

PLP - Recuperatorio del Primer Parcial - 2^{do} cuatrimestre de 2022

Este examen se aprueba obteniendo al menos dos ejercicios bien menos (B-) y uno regular (R). Las notas para cada ejercicio son: -, I, R, B-, B. Poner nombre, apellido y número de orden en todas las hojas, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.
El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolverlo.

Ejercicio 1 - Programación funcional

Durante este ejercicio **no** se puede usar recursión explícita, a menos que se indique lo contrario. Para resolver un ítem pueden utilizarse las funciones definidas en los ítems anteriores, más allá de si fueron resueltos correctamente o no. Dar el tipo de todas las funciones pedidas.

En este ejercicio trabajaremos con matrices infinitas. Se define el tipo `Matriz` de la siguiente manera:

```
data Matriz a = NuevaMatriz a | Agregar a Int Int (Matriz a)
```

Donde `NuevaMatriz v` representa una matriz con valor por defecto `v`, y `Agregar v x y m` representa la matriz que resulta de agregar el valor `v` en la fila `x` y columna `y` a la matriz `m`. El valor de una posición de la matriz será el último que se haya agregado en la fila y columna correspondientes, o el valor por defecto si nunca se agregó un valor en esa posición.

Utilizaremos las siguientes matrices para los ejemplos:

```
m1 = Agregar 5 1 2 $ Agregar 2 2 1 $ Agregar 3 1 1 $ NuevaMatriz 0
```

```
m2 = Agregar 'a' 1 2 $ Agregar 'b' 1 2 $ Agregar 'c' 1 1 $ NuevaMatriz 'd'
```

```
m3 = Agregar 0 1 2 $ Agregar 4 0 0 $ Agregar 2 1 1 $ NuevaMatriz 1
```

- a) Definir el esquema de recursión estructural `foldMatriz` para este tipo de matrices, y dar su tipo. En este punto se permite usar recursión explícita.
- b) Definir la función `ver :: Int -> Int -> Matriz a -> a`, que devuelva el valor de la posición de una matriz correspondiente a la fila y columna indicadas.
Por ejemplo: `ver 1 1 m2` devuelve 'c'. `ver 1 2 m2` devuelve 'a'. `ver 0 0 m2` devuelve 'd'.
- c) Definir `suma :: Num a => Matriz a -> Matriz a -> Matriz a`, que calcule la suma de dos matrices.
Por ejemplo: `suma m1 m3` es la matriz con 4 en la posición (0,0), 5 en las posiciones (1,1) y (1,2), 3 en (2,1) y 1 en todas las demás (porque $0 + 1 = 1$).
Se recomienda definir `suma` por recursión (`foldMatriz`) en la primera matriz que toma como parámetro. Pensar bien cuál es el tipo que va a devolver el fold, y cuál es el resultado de la recursión.
Sugerencia: definir primero el `map` para matrices.
- d) Definir la matriz `matrizDeSumas`, que contenga en cada posición con índices naturales la suma de los índices de su fila y columna. El valor por defecto no se usa, por lo que puede ser 0 o un error. Puede colgarse si se intenta ver un valor con índices negativos.
Por ejemplo: `ver 2 3 matrizDeSumas` devuelve 5.
Sugerencia: usar una lista de pares como estructura intermedia.

Ejercicio 2 - Inferencia de tipos Considerar el Cálculo Lambda tipado extendido con listas. Se extiende el cálculo para poder construir listas a partir de un primer elemento, una función generadora y una condición de corte.

El conjunto de tipos no se modifica. El conjunto de términos se extiende de la siguiente manera:

$$M ::= \dots \mid \text{from } M_1 \text{ until}_x M_2 \text{ by } M_3$$

Una expresión de la forma `from M_1 untilx M_2 by M_3` , donde M_1 es un elemento cualquiera, M_2 es una expresión booleana que puede tener libre la variable x , y M_3 una función que se aplicará a cada elemento

para generar el siguiente, reducirá a una lista cuyo primer elemento - de haberlo - es el valor de M_1 , y cada elemento subsiguiente se obtiene aplicando M_3 al elemento anterior hasta que se satisfaga la condición M_2 tomando como x al elemento actual. Si M_2 es verdadera al tomar M_1 como x , entonces la lista resultante será vacía.

Se incorpora la siguiente regla de tipado:

$$\frac{\Gamma \triangleright M_1 : \sigma \quad \Gamma, x : \sigma \triangleright M_2 : \text{Bool} \quad \Gamma \triangleright M_3 : \sigma \rightarrow \sigma}{\Gamma \triangleright \text{from } M_1 \text{ until}_x M_2 \text{ by } M_3 : [\sigma]} \text{(T-FUB)}$$

- Extender el algoritmo de inferencia para soportar la nueva extensión (la extensión original para listas se considera dada).
- Aplicar el algoritmo extendido con el método del árbol para dar el tipo de la siguiente expresión, exhibiendo las sustituciones utilizadas. De no tipar, indicar el motivo.

$\text{from } x \text{ until}_x \text{ if isZero}(x) \text{ then False else True by } \lambda x. \text{Succ}(x)$

Ejercicio 3 - Subtipado

Se extiende el cálculo lambda con el tipo $\text{AHD}_{\sigma, \tau}$, que representa un árbol binario no vacío cuyos nodos internos pueden tener datos de un tipo diferente al de sus hojas. (AHD = árbol con hojas distinguidas).

$\sigma ::= \dots \mid \text{AHD}_{\sigma, \sigma}$

$M ::= \text{hoja}_{\sigma, \tau}(M) \mid \text{rama}(M, M) \mid \text{bin}(M, M, M) \mid \text{fold } M \text{ hoja}_x \rightsquigarrow M; \text{rama}_{n, \text{rec}} \rightsquigarrow M; \text{bin}_{ri, v, rd} \rightsquigarrow M$
donde:

- $\text{AHD}_{\sigma, \tau}$ es el tipo de los árboles con nodos internos de tipo σ y hojas de tipo τ .
- $\text{hoja}_{\sigma, \tau}(M)$ representa al $\text{AHD}_{\sigma, \tau}$ cuyo único elemento es M (es una hoja).
- $\text{rama}(M, N)$ describe al árbol con un nodo interno M y un subárbol N .
- $\text{bin}(M, N, O)$ representa al árbol con raíz N y subárboles M y O .
- $\text{fold } M \text{ hoja}_x \rightsquigarrow N; \text{rama}_{n, \text{rec}} \rightsquigarrow O; \text{bin}_{ri, v, rd} \rightsquigarrow P$ es el esquema de recursión estructural para árboles con hojas distinguidas, donde las variables x, n, rec, ri, v y rd se ligarán a los valores correspondientes según la estructura del árbol M .

Las reglas de tipado son las siguientes:

$$\frac{\Gamma \triangleright M : \tau}{\Gamma \triangleright \text{hoja}_{\sigma, \tau}(M) : \text{AHD}_{\sigma, \tau}} \text{(T-HOJA)} \quad \frac{\Gamma \triangleright M : \sigma \quad \Gamma \triangleright N : \text{AHD}_{\sigma, \tau}}{\Gamma \triangleright \text{rama}(M, N) : \text{AHD}_{\sigma, \tau}} \text{(T-RAMA)}$$

$$\frac{\Gamma \triangleright M : \text{AHD}_{\sigma, \tau} \quad \Gamma \triangleright N : \sigma \quad \Gamma \triangleright O : \text{AHD}_{\sigma, \tau}}{\Gamma \triangleright \text{bin}(M, N, O) : \text{AHD}_{\sigma, \tau}} \text{(TBIN)}$$

$$\frac{\Gamma \triangleright M : \text{AHD}_{\sigma, \tau} \quad \Gamma, x : \tau \triangleright N : \rho \quad \Gamma, n : \sigma, \text{rec} : \rho \triangleright O : \rho \quad \Gamma, ri : \rho, v : \sigma, rd : \rho \triangleright P : \rho}{\Gamma \triangleright \text{fold } M \text{ hoja}_x \rightsquigarrow N; \text{rama}_{n, \text{rec}} \rightsquigarrow O; \text{bin}_{ri, v, rd} \rightsquigarrow P : \rho} \text{(T-FOLD AHD)}$$

- Dar la(s) regla(s) de subtipado para esta extensión y justificar en términos del principio de substitutividad.
- Utilizando las reglas de tipado y subtipado, derivar el siguiente juicio de tipado:

$\emptyset \triangleright (\lambda a : \text{AHD}_{\text{Int}, \text{Nat}}. \text{fold } a \text{ hoja}_x \rightsquigarrow \text{True}; \text{rama}_{n, \text{rec}} \rightsquigarrow 0; \text{bin}_{ri, v, rd} \rightsquigarrow \text{Succ}(ri))$
 $(\text{rama}(0, \text{hoja}_{\text{Nat}, \text{Bool}}(\text{True}))) : \text{Int}$